

Overview

The N64 only has access to the cartridge slot (known as the PI bus). In order to allow the bootloader and menu programs to access the cartridge hardware, and allow homebrew programmers the ability to use the special features of the cartridge (such as CompactFlash and USB), an interface was developed.

All the hardware on the circuit board is accessible by means of a field-programmable gate array. Automatically configured on every power-on, the FPGA's job is to coordinate communication between various components on the 64drive PCB such as SDRAM, USB, the PI bus, and also emulates all N64 save types.

Modes of Operation

The 64drive can be operated in one of two modes at a time.

MENU MODE Default mode upon power-up. In this mode, when the N64 is powered on, it receives the bootloader, which searches the memory card for MENU.BIN and loads it.

DEV MODE This mode is switched to once a USB cable is plugged in. In this mode, the bootloader is ignored. It is intended that the user upload an image to SDRAM using PC-side software prior to the N64 booting.

Save writeback on reset is disabled in this mode.
Also allows for debugging.

CI (Cartridge Interface)

The 'CI' interface (cart interface) is a small range of memory in the upper area of the PI address space that contains registers and some block memory that the FPGA handles.

Registers

Address	Size	Function	Read/Write
0x1800 0000	512 bytes	Buffer (256 words)	R/W
0x1800 0200	4 bytes	Status Register	Read
0x1800 0208	4 bytes	Command Register	Write
0x1800 0210	4 bytes	LBA Register	Write
0x1800 0218	4 bytes	Length Register	R/W
0x1800 02F4	4 bytes	Persistent Variable Storage	R/W
0x1800 02F8	4 bytes	Button Register	Read
0x1800 02FC	4 bytes	Revision Register	Read
0x1800 1000	2048 bytes	EEPROM contents (1024 words)	R/W
0x1800 1800	1024 bytes	Save Writeback LBA List (256 dwords)	R/W

To perform a command:

1. Check the status register to make sure CI is not busy
2. Write all necessary parameters
3. Write the command code to the command register

Status Register Table

The status register is 16 bits repeated twice.

1 st byte	3 rd nibble	4 th nibble
0x00 Ready	0x0 USB TX empty	0x0 USB RX empty
0x10 CI Busy	0xF USB TX non-empty	0xF USB RX non-empty

Example: 0x10FF

Meaning: 0x10 – CI Busy

0xF – There is data sent, waiting on the PC to read it

0xF – There is data waiting for us in the receive buffer

CI Commands

Command	ID	Parameters
READ SECTOR INTO BUFFER	0x01	LBA
READ SECTOR TO SDRAM	0x02	LBA, RAMADDR
READ MULTIPLE SECTORS TO SDRAM	0x03	LBA, RAMADDR, NUM_SEC
WRITE BUFFER INTO SECTOR	0x10	LBA
READ USB TO BUFFER	0x30	-
WRITE BUFFER TO USB	0x31	LENGTH
SET SAVE TYPE	0xD0	SAVE_TYPE
DISABLE BYTESWAP ON LOAD	0xE0	-
ENABLE BYTESWAP ON LOAD	0xE1	-
ENABLE CARTROM WRITES	0xF0	-
DISABLE CARTROM WRITES	0xF1	-
SET CYCLE TIME	0xFD	NUM_CYCLES

Command Listing

READ SECTOR INTO BUFFER

Copies one sector into the on-chip 512byte buffer.

0x1800 0210 [LBA REG]	LBA (sector number)
-----------------------	---------------------

READ SECTOR TO SDRAM

Copies 512 bytes from CompactFlash to an address in SDRAM.

0x1800 0210 [LBA REG]	LBA (sector number)
0x1800 0004	SDRAM address (0x0 - 0x1FFFFFFF)

READ MULTIPLE SECTORS TO SDRAM

Copies any number of sectors from CF to an address in SDRAM.

0x1800 0210 [LBA REG]	LBA (sector number)
0x1800 0218 [LENGTH REG]	Number of sectors
0x1800 0004	SDRAM address (0x0 - 0x1FFFFFFF)

WRITE SECTOR FROM BUFFER

Writes the on-chip 512byte buffer to the CF sector specified in the LBA register.

0x1800 0210 [LBA REG]	LBA (sector number)
-----------------------	---------------------

READ USB INTO BUFFER

Reads in a maximum of 512 bytes from the USB chip FIFO. The number of bytes is written to the length register for the information of the user. [NOTE: As of firmware 1.00 the length read is fixed at 512 bytes!]

If there is more data still in the FIFO, the CI status register will reflect that, and you should perform this command again until it is emptied.

If the FIFO is exhausted while it is read into the buffer, then the length register will reflect the amount it was able to read.

Before issuing this command, please confirm that there is actually data in the FIFO by reading the status register.

0x1800 0218 [LENGTH REG]	Amount of data read (words) [256 maximum]
--------------------------	---

WRITE BUFFER TO USB

Writes a maximum of 512 bytes from the buffer to the USB chip FIFO.

If you wish to send more than 512 bytes, you will need to rewrite the buffer with more data and call this command again.

Before issuing this command, please confirm that the USB TX buffer is empty. Since the USB TX buffer is bigger than 512 bytes, you may decide to write more data before polling the status, at your own discretion.

0x1800 0218 [LENGTH REG]	Amount to write to USB (words) [256 maximum]
--------------------------	--

SET SAVE TYPE

Sets the type of save support the FPGA should handle.

0x1800 0000	Save type: 0 = none 1 = EEPROM 4Kbit 2 = EEPROM 16Kbit 3 = SRAM 256Kbit 4 = FlashRAM 1Mbit 5 = SRAM 768Kbit (for Dezaemon 3D)
-------------	---



WARNING

Simply enabling saving without writing valid LBAs to the writeback LBA list buffer will cause catastrophic corruption of the CF card because of sector writes to undefined LBAs.

SET CYCLE TIME

Sets the number of clock cycles of the OE and WE pulse widths for the CompactFlash interface.

Currently the logic runs at 50MHz so the cycle time is 20ns. Setting this value too low will cause data corruption during reading and writing.

0x1800 0000	Clock cycles
-------------	--------------

DISABLE BYTESWAP ON LOAD

Sets normal byte ordering (Z64).

ENABLE BYTESWAP ON LOAD

Causes all words transferred directly from CF to SDRAM to be byteswapped, for loading V64 format images. This has no effect on copying from the buffer to SDRAM. The default is `DISABLED`.

ENABLE CARTROM WRITES

Normally, reads from cartridge space (`0x1000 0000-0x1400 0000`) are mapped to SDRAM. This command lets writes fall through to SDRAM as well. The default is `DISABLED`.

DISABLE CARTROM WRITES

Locks cartridge ROM (SDRAM) against direct modification by the N64. This is the default state.

Additional Registers

In addition to the registers used for commands, there are a few more that are of importance.

REVISION REGISTER

Read from this location to get the FPGA configuration revision number. The result is a 32-bit integer with identical 16-bit halves. To get a meaningful number, AND the value with `0xFFFF` to get the relevant 16 bits. For example, on the first revision code, the value is decimal 100, which means revision 1.00.

This information is displayed in the About tab of the menu.

PERSISTENT VARIABLE STORAGE

This register is just 4 bytes of storage that can be used for any purpose. The bootloader writes its version number to this register so that the menu can read it later on.

EEPROM CONTENTS

This is a 2048-byte (1024 word) buffer that stores the contents of the EEPROM chip being emulated. The menu writes to this when loading an image if that image has EEPROM save data associated with it.

If the SET SAVE TYPE command was executed with the parameter '1', the valid size is 512 bytes. If it was executed with '2', the valid size is 2048 bytes.



WARNING

It is not recommended to modify the buffer contents while the EEPROM is being accessed by the N64.

SAVE WRITEBACK LBA LIST

This is a 1024-byte (256 dword) buffer that stores a list of LBAs occupied by the image's save file. The LBAs are stored as 32bits each, even though 27bit LBA addressing is used.

If the save type is EEPROM 4k/16k, the on-FPGA blockram buffer is copied to these LBAs. If the save type is SRAM, data is copied from 0x1FFC000 of SDRAM (word addressed). If the save type is FLASHRAM, data is copied from 0x1FF0000 of SDRAM (word addressed).

The save writeback state machine is activated by the NMI signal on the cartridge slot. The user must press and release the reset button on the N64 to trigger a NMI.



WARNING

It is the programmer's responsibility to write the correct LBA values in this register whenever saving is enabled. The LBAs are undefined upon power up and simply enabling saving will cause catastrophic corruption of the CF card.

Implementation Notes

Polling the FPGA Status

It is important to check the status of the FPGA state machine before sending any commands, to avoid interfering with its operations. Since most operations take some time to complete, you must check for completion.

Some older CompactFlash devices and mechanical memories such as microdrives may have latencies up to several seconds (for example, if the microdrive has spun down and must spool up again).

```
void ciWait()
{
    long timeout = 0;
    char buf[4];

    // poll the status register, and look at bits [15:8]
    do{
        osPiReadIo(0xB8000200, buf); while(osPiGetStatus() != 0);
        timeout++;
        if(timeout == 2000000){
            // set error condition here
            return;
        }
    }while(buf[2] != 0);
}
```

Transferring Data to SDRAM

```
void ciSectorsToRam(int ramaddr, int lba, int sectors)
{
    ciWait();

    // write LBA to the fpga register
    osPiWriteIo(0xB8000210, lba ); while(osPiGetStatus() != 0);
    osPiWriteIo(0xB8000004, ramaddr); while(osPiGetStatus() != 0);
    osPiWriteIo(0xB8000008, sectors); while(osPiGetStatus() != 0);

    // write "read sectors to ram" command
    osPiWriteIo(0xB8000208, 0x3); while(osPiGetStatus() != 0);

    ciWait();
}
```

Reading to a Buffer

```
void ciReadSector(unsigned char *buf, int lba)
{
    ciWait();

    // write LBA to the fpga register
    osPiWriteIo(0xB8000210, lba); while(osPiGetStatus() != 0);
    // write "read sector" command
    osPiWriteIo(0xB8000208, 1); while(osPiGetStatus() != 0);
    ciWait();
}
```

```
// DANGER WILL ROBINSON
// The R4300's data cache is about to be invalidated by the DMA.
// Write back the DCACHE to prevent inconsistencies.
osWritebackDCacheAll();

// read the 512-byte onchip buffer
osPiRawStartDma(OS_READ, 0xB8000000, (u32)buf, 512);
while(osPiGetStatus() != 0);
}
```

USB Interface

The USB interface is used for communication to a program running on the PC.

Since the USB implementation in hardware is simply an 8-bit wide dumb bus, it's up to the programmer to come up with a protocol for sending data between the N64 and PC.

The packet structure for writing from PC to N64 as supported by the FPGA is as follows.

All commands with a first byte $\geq 0x80$ are immediate commands that have no data following them, and are processed immediately.

All commands with a first byte $< 0x80$ are transfer commands, and should be immediately followed with the exact amount of data specified in the command packet.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
BULK WRITE TO SDRAM	10	0A	Length (words) [1]				SDRAM Address [2]									
	20	0A	Length (words) [1]				SDRAM Address [2]									
BULK ERASE CONFIG	80	04	BE	EF												
WRITE TO CONFIG	70	0A	Length (bytes)				Offset (bytes)									
WRITE TO N64 BUS	90	08	Address				Data									
WRITE TO N64 BUS LONG	91	08	Address				Data									
SET SAVE TYPE	D0	04	[3]	00												

All fields are in hexadecimal
All multi-byte values are big-endian

Notes

- [1] A word is 16 bits.
- [2] SDRAM is word-addressed, for example an offset of 8 kilobytes is 4,096 words.
- [3] Save type. Refer to the CI section for available save modes.

Revision Information

October 2, 2010

First internal draft

June 25, 2011

Current internal revision

Copyright Information

© 2010-2011 Marshall H / Retroactive

<http://64drive.retroactive.be/>

"Nintendo" is a registered trademark of Nintendo of America Inc.

Nintendo 64 is a registered trademark of Nintendo Company, Limited

This product is not endorsed or supported by Nintendo. Use the device at your own risk.

To submit any inaccuracies or inconsistencies in this document, please e-mail support at the above web address.